# 图像内插

# Image Interpolation

*三种传统内插方法介绍与解析*

丁昊

August 2016

**摘要**

内插是在诸如放大、收缩、旋转和几何校正等任务中广泛应用的基本工具。放大图像（或称为上采样（upsampling）或图像插值（interpolating））的主要目的是放大原图像, 从而可以显示在更高分辨率的显示设备上。图像放大几乎都是采用内插值方法，即在原有图像像素的基础上在像素点之间采用合适的插值算法插入新的元素。

# 目录


**1 基本概念** **1**

**2 分类** **2**

**3 最近邻内插** **3**

3.1 代码 . . . . . 3

3.2 结果 . . . . . 5

**4 双线性内插** **6**

4.1 原理 . . . . . 6

4.2 代码 . . . . . 7

4.3 结果 . . . . . 9

**5 双三次内插** **10**

5.1 原理 . . . . . 10

5.2 代码 . . . . . 11

5.3 结果 . . . . . 13

# 第 1 章

## 基本概念

内插是一个通过已知的离散数据求未知数据的过程，我所研究的这一部分是利用内插方法来放大一个图像。

对于一副原图像，假设我们想将它放大到原来大小的 2 倍，那么设想一下，因为每个像素点的大小是相同的，在长和宽变为两倍后，像素点的个数自然变为了原来的 4 倍。如果我们想使放大后的图像尽量的不失真，就需要通过某种方法，根据原图仅有的像素点来求得放大后图像中每个像素点的值。这个过程就是内插。

拿最近邻内插来解释说明。我们先预设一个原来两倍大小的图像，将其等比例对应到原图的每个像素点。当然，因为新图像素点的个数为原图的四倍，所以对应的相同位置上原图的一个像素点就对应了新图的四个像素点。这时，使新图的每个像素点都等于原图对应位置的像素值，就完成了最近邻内插过程。我举得这个例子是正好成倍数的，绝大部分情况并非这样，但原理相同，即将对应位置最近邻的一个点的值赋给新图。

# 第 2 章

## 分类

- 传统插值：最近邻内插，双线性内插，三线性内插。也是 DIP 课本上介绍的三类内插。
- 给予边缘的图像插值算法：基于原始低分辨图像边缘的方法和基于插值后高分辨率图像边缘的方法。基于原始低分辨率图像边缘的方法: 首先检测低分辨率图像的边缘，然后根据检测的边缘将像素分类处理，对于平坦区域的像素，采用传统方法插值；对于边缘区域的像素，设计特殊插值方法，以达到保持边缘细节的目的。基于插值后高分辨率图像边缘的方法这类插值方法: 首先采用传统方法插值低分辨率图像，然后检测高分辨率图像的边缘，最后对边缘及附近像素进行特殊处理，以去除模糊，增强图像的边缘。
- 给予区域的图像插值算法：首先将原始低分辨率图像分割成不同区域，然后将插值点映射到低分辨率图像，判断其所属区域，最后根据插值点的邻域像素设计不同的插值公式，计算插值点的值。
- 偏微分方程插值 (PDE), 分形，小波逆向插值这三种也是插值算法的主流之一。小波与分形算法计算复杂度高，效果较好，小波边缘处理最好，分形次之。

# 第 3 章

# 最近邻内插

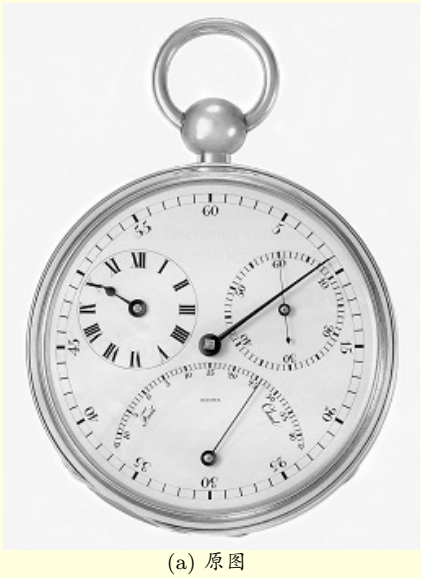
最近邻内插即在基本概念中所解释的过程，取的是对应点最近邻的值。

## 3.1 代码

```
1
2 #include <iostream>
3 #include <opencv2/imgproc/imgproc.hpp>
4 #include <opencv2/highgui/highgui.hpp>
5 #include <math.h>
6
7 using namespace cv;
8 using namespace std;
9
10    int main(int argc, char* argv[])
11    {
12  //放大倍数
13  double x=2;
14
15        //读图语句报错
16  if ( argc != 2 )
17  {
18    cout<<"Wrong arguments."<<endl;
19    cout<<"Usage:"<<endl;
20    cout<<"\t "<<argv[0]<<" Image"<<endl;
21    exit(7);
22  }
23
24  string image = argv[1];
25        Mat src = imread(image);
26
27  //读图格式报错
28  if( !src.data )
29  {
30    cout<<"Can't process this kind of image"<<endl;
31    exit(8);
32  }
33
```

```
34          //显示
35          imshow("src",src);
36
37          //处理长宽
38      int row=src.rows;
39      int col=src.cols;
40      float nrow=x*(float)row;
41      float ncol=x*(float)col;
42
43      //创建矩阵
44          Mat dst = Mat::zeros(nrow,ncol,CV_8UC3);
45          int X=0;
46      int Y=0;
47
48      //最邻近内插
49      for ( int i = 0; i < nrow; i++)
50      {
51        for(int j = 0; j < ncol; j++)
52        {
53          X= cvRound ( i / (double)x );
54          Y= cvRound ( j / (double)x );
55          if( X < row && X ≥ 0 && Y ≥ 0 && Y ≤ col )
56          {
57            dst.at<cv::Vec3b>(i,j)[0]=src.at<cv::Vec3b>(X,Y)[0];
58            dst.at<cv::Vec3b>(i,j)[1]=src.at<cv::Vec3b>(X,Y)[1];
59            dst.at<cv::Vec3b>(i,j)[2]=src.at<cv::Vec3b>(X,Y)[2];
60          }
61        }
62      }
63
64
65
66          imshow("dst",dst);
67      imwrite("nearest.bmp",dst);
68          waitKey(0);
69          return 0;
70        }
```

### 3.2 结果



(a) 原图

(b) 效果图

图 1: 最近邻内插

# 第 4 章

# 双线性内插

## 4.1 原理

双线性内插的赋值由下面公式得到：

$f(x,y) \approx f(0,0)(1-x)(1-y) + f(1,0)x(1-y) + f(0,1)(1-x)y + f(1,1)xy$

其中 f(0,0)、f(1,0)、f(0,1)、f(1,1) 四个点为目标图像像素点对应到原图的四个最近邻点，x、y 的值并非绝对坐标而是相对坐标，即 0 到 1 之间的小数。将上式化简即可得到书上的公式：

$v(x,y) = ax + by + cxy + d$

而四个系数的取值此时为：

- a=f(1,0)-f(0,0)
- b=f(0,1)-f(0,0)
- c=f(1,1)-f(0,1)-f(1,0)+f(0,0)
- d=f(0,0)

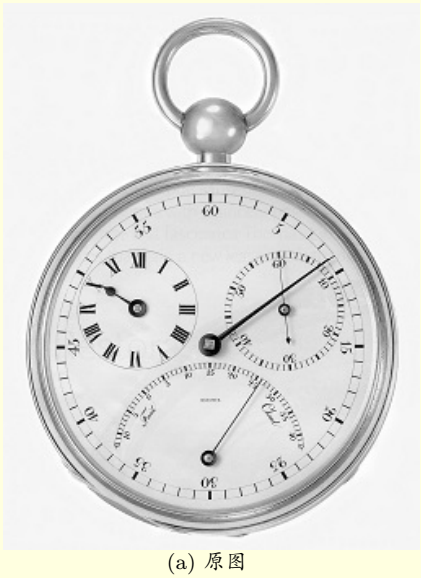
然而，用这种取系数的方法会导致输出图像部分溢出，所以不推荐使用。

## 4.2 代码

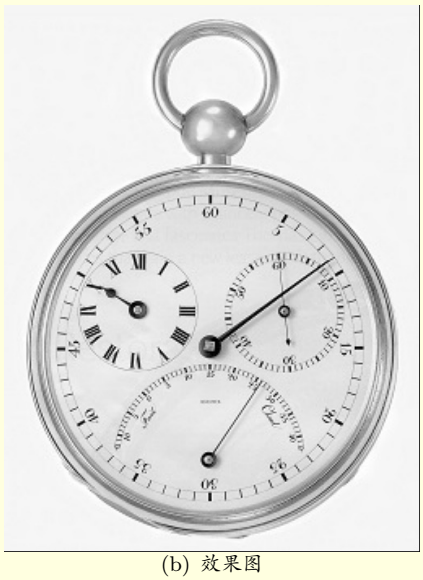
```
1
2  #include <iostream>
3  #include <opencv2/imgproc/imgproc.hpp>
4  #include <opencv2/highgui/highgui.hpp>
5  #include <math.h>
6
7  using namespace cv;
8  using namespace std;
9
10     int main(int argc, char* argv[])
11     {
12   //放大倍数
13   double x=2;
14
15         //读图语句报错
16   if ( argc != 2 )
17   {
18     cout<<"Wrong arguments."<<endl;
19     cout<<"Usage:"<<endl;
20     cout<<"\t "<<argv[0]<<" Image"<<endl;
21     exit(7);
22   }
23
24   string image = argv[1];
25         Mat src = imread(image);
26
27   //读图格式报错
28   if( !src.data )
29   {
30     cout<<"Can't process this kind of image"<<endl;
31     exit(8);
32   }
33
34         //显示
35         imshow("src",src);
36
37         //处理长宽
38   int row=src.rows;
39   int col=src.cols;
40   float nrow=x*(float)row;
41   float ncol=x*(float)col;
42
43   //创建矩阵
```

```
44          Mat dst = Mat::zeros(nrow,ncol,CV_8UC3);
45
46      float X=0,Y=0;
47      CvScalar a,b,c,d;
48
49      //双线性内插
50      for ( int i = 0; i < nrow-1; i++)
51      {
52        for(int j = 0; j < ncol-1; j++)
53        {
54          X= i / (double)x ;
55          Y= j / (double)x ;
56          float ux=(int)X,uy=(int)Y;
57          ux=abs(ux-X);uy=abs(uy-Y);
58
59          if( X < row-1 && X ≥ 0 && Y ≥ 0 && Y < col-1 )
60          {
61        //取四个点
62          a = src.at<cv::Vec3b>((int)X,(int)Y);
63          b = src.at<cv::Vec3b>((int)X,(int)Y+1);
64          c = src.at<cv::Vec3b>((int)X+1,(int)Y);
65          d = src.at<cv::Vec3b>((int)X+1,(int)Y+1);
66          }
67        //系数相加为处理后的点值
68            dst.at<cv::Vec3b>(i,j)[0]=(a.val[0]*(1-ux)+c.val[0]*ux)*(1-uy)+(b.val[0]*(1-
        ux)+d.val[0]*ux)*uy;
69            dst.at<cv::Vec3b>(i,j)[1]=(a.val[1]*(1-ux)+c.val[1]*ux)*(1-uy)+(b.val[1]*(1-
        ux)+d.val[1]*ux)*uy;
70            dst.at<cv::Vec3b>(i,j)[2]=(a.val[2]*(1-ux)+c.val[2]*ux)*(1-uy)+(b.val[2]*(1-
        ux)+d.val[2]*ux)*uy;
71
72        }
73      }
74
75
76
77          imshow("dst",dst);
78          imwrite("double.bmp",dst);
79
80          waitKey(0);
81          return 0;
82      }
```

### 4.3 结果



(a) 原图



(b) 效果图

图 2: 双线性内插

# 第 5 章

# 双三次内插

## 5.1 原理

双三次插值通过下式进行计算：

$f(x,y) = a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{21}x^2y + a_{12}xy^2 + a_{22}x^2y^2 + a_{30}x^3 + a_{03}y^3 + a_{31}xy^3 + a_{32}x^3y^2 + a_{23}x^2y^3 + a_{33}x^3y^3$

也就是：

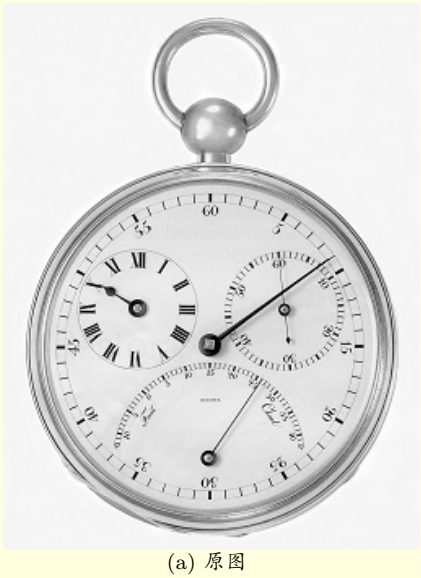
$f(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^iy^j$

计算系数 $a_{ij}$ 的取值依赖于插值数据的特性。如果已知插值函数的导数，常用的方法就是使用四个顶点的高度以及每个顶点的三个导数。一阶导数 $h'x$ 与 $h'y$ 表示 x、y 放心的表面斜率，二阶相互导数 $h''xy$ 表示同时在 x 与 y 方向的斜率。这些值可以通过分别连续对 x 与 y 向量取微分得到。对于网格单元的每个顶点，将局部坐标 (0,0)、(1,0)、(0,1) 和 (1,1) 带入这些方程，再解这 16 个方程。

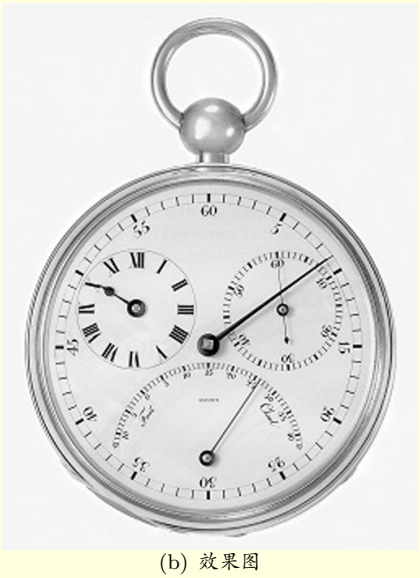
## 5.2 代码

```
#include <iostream>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <math.h>

using namespace cv;
using namespace std;

int main(int argc, char* argv[])
    {
        //导入图像
  if ( argc != 2 )
  {
    cout<<"Wrong arguments."<<endl;
    cout<<"Usage:"<<endl;
    cout<<"\t "<<argv[0]<<" Image"<<endl;
    exit(7);
  }


  string image = argv[1];
        Mat src = imread(image);

      //读图格式报错
  if( !src.data )
  {
    cout<<"Can't process this kind of image"<<endl;
    exit(8);
  }

        //显示
        imshow("src",src);

  Mat dst;
  resize(src,dst,Size(),2,2,INTER_CUBIC);


        imshow("dst",dst);
      imwrite("nearest-resize.bmp",dst);
        waitKey(0);
        return 0;
    }
```

## 5.3 结果



(a) 原图



(b) 效果图

图 3: 双三次内插